Exercise 1: Control Structures

Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60 years old.

o Question: Write a PL/SQL block that loops through all

customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

Scenario 2: A customer can be promoted to VIP status based on their balance.

○ Question: Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

Scenario 3: The bank wants to send reminders to customers whose loans are due within the next 30 days.

Question: Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE loans';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE customers';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

age NUMBER,

balance NUMBER,

IsVIP VARCHAR2(5)

);

CREATE TABLE loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

interest\_rate NUMBER,

due\_date DATE,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

INSERT INTO customers VALUES (1, 'Alice Smith', 65, 12000, 'FALSE');

INSERT INTO customers VALUES (2, 'Bob Jones', 55, 8000, 'FALSE');

INSERT INTO customers VALUES (3, 'Charlie Brown', 70, 15000, 'FALSE');

INSERT INTO customers VALUES (4, 'Diana Prince', 40, 9500, 'FALSE');

INSERT INTO loans VALUES (101, 1, 5.5, SYSDATE + 10);

INSERT INTO loans VALUES (102, 2, 6.0, SYSDATE + 40);

INSERT INTO loans VALUES (103, 3, 4.8, SYSDATE + 20);

INSERT INTO loans VALUES (104, 4, 5.2, SYSDATE + 5);

COMMIT;

BEGIN

FOR rec IN (

SELECT c.customer\_id, c.name, l.loan\_id

FROM customers c

JOIN loans l ON c.customer\_id = l.customer\_id

WHERE c.age > 60

) LOOP

UPDATE loans

SET interest\_rate = interest\_rate - 1

WHERE loan\_id = rec.loan\_id;

DBMS\_OUTPUT.PUT\_LINE('Updated loan-ID ' || rec.loan\_id ||

',for customer ' || rec.name ||

' by 1% discount on interest rate.');

END LOOP;

COMMIT;

END;

/

BEGIN

FOR rec IN (

SELECT customer\_id, name, balance FROM customers WHERE balance > 10000

) LOOP

UPDATE customers

SET IsVIP = 'TRUE'

WHERE customer\_id = rec.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('Customer ID:' || rec.customer\_id ||

',' || rec.name ||

' promoted to VIP.');

END LOOP;

COMMIT;

END;

/

BEGIN

FOR rec IN (

SELECT c.customer\_id, c.name, l.loan\_id, l.due\_date

FROM customers c

JOIN loans l ON c.customer\_id = l.customer\_id

WHERE l.due\_date BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder Loan - ID: ' || rec.loan\_id ||

', Customer ID ' || rec.customer\_id ||

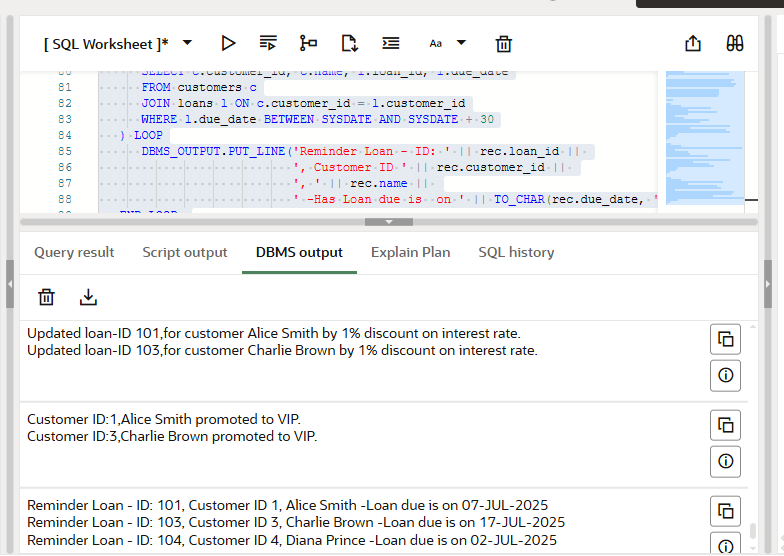
', ' || rec.name ||

' -Has Loan due is on ' || TO\_CHAR(rec.due\_date, 'DD-MON-YYYY'));

END LOOP;

END;

/



Exercise 3: Stored Procedures

Scenario 1: The bank needs to process monthly interest for all savings accounts.

○ Question: Write a stored procedure

ProcessMonthlyInterest that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

Scenario 2: The bank wants to implement a bonus scheme for employees based on their performance.

o Question: Write a stored procedure UpdateEmployeeBonus that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

Scenario 3: Customers should be able to transfer funds between their accounts.

○ Question: Write a stored procedure TransferFunds that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

BEGIN

DBMS\_OUTPUT.PUT\_LINE('DBMS Output is active...');

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE accounts';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE employees';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

account\_type VARCHAR2(20),

balance NUMBER

);

CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

department VARCHAR2(50),

salary NUMBER

);

INSERT INTO accounts VALUES (1, 101, 'savings', 1000);

INSERT INTO accounts VALUES (2, 102, 'checking', 2000);

INSERT INTO accounts VALUES (3, 101, 'savings', 1500);

INSERT INTO accounts VALUES (4, 103, 'savings', 3000);

INSERT INTO employees VALUES (1, 'John Smith', 'Sales', 5000);

INSERT INTO employees VALUES (2, 'Jane Doe', 'HR', 4500);

INSERT INTO employees VALUES (3, 'Bob Lee', 'Sales', 5200);

COMMIT;

/

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

v\_new\_balance NUMBER;

BEGIN

FOR acc IN (

SELECT account\_id, balance FROM accounts WHERE LOWER(account\_type) = 'savings'

) LOOP

v\_new\_balance := acc.balance \* 1.01;

UPDATE accounts SET balance = v\_new\_balance WHERE account\_id = acc.account\_id;

DBMS\_OUTPUT.PUT\_LINE('Interest applied - Account ID ' || acc.account\_id ||

', Current Balance = $' || TO\_CHAR(v\_new\_balance, 'FM9999.00'));

END LOOP;

COMMIT;

END;

/

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

v\_new\_salary NUMBER;

BEGIN

FOR emp IN (

SELECT employee\_id, name, salary FROM employees WHERE department = p\_department

) LOOP

v\_new\_salary := emp.salary + (emp.salary \* p\_bonus\_percent / 100);

UPDATE employees SET salary = v\_new\_salary WHERE employee\_id = emp.employee\_id;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied - ' || emp.name ||

' (ID: ' || emp.employee\_id || '), New Salary = $' ||

TO\_CHAR(v\_new\_salary, 'FM99999.00'));

END LOOP;

COMMIT;

END;

/

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT balance INTO v\_balance FROM accounts WHERE account\_id = p\_from\_account;

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance in source account.');

END IF;

UPDATE accounts SET balance = balance - p\_amount WHERE account\_id = p\_from\_account;

UPDATE accounts SET balance = balance + p\_amount WHERE account\_id = p\_to\_account;

DBMS\_OUTPUT.PUT\_LINE('Transferred $' || p\_amount ||

' from Account ID ' || p\_from\_account ||

' to Account ID ' || p\_to\_account);

COMMIT;

END;

/

BEGIN

ProcessMonthlyInterest;

UpdateEmployeeBonus('Sales', 10);

TransferFunds(4, 2, 500);

END;

/

